jQuery biblioteka

Biblioteku jQuery je prvobitno kreirao John Resig u januaru 2006, pod uticajem ranije popularne biblioteke cssQuery, koju je kreirao Dean Edwards.

Ova biblioteka omogućava jednostavnije pisanje JavaScript koda, odnosno olakšava:

- HTML/DOM manipulaciju;
- CSS manipulaciju;
- manipulaciju događajima;
- rad sa efektima i animacijama;
- AJAX.

jQuery biblioteku koristi 73% od deset miliona najpopularnijih sajtova. Web analiza pokazuje da je to najrasprostranjenija JavaScript biblioteka, sa tri do četiri puta većom upotrebom od bilo koje druge JavaScript biblioteke.

U ovoj lekciji ćemo obraditi osnove jQuery biblioteke, implementaciju, selektore, manipulaciju DOM-om, upravljanje događajima i slanje i obradu zahteva ka serveru.

Dodavanje jQuery biblioteke u projekat obavlja se na dva načina:

- preuzimanjem jQuery biblioteke sa zvaničnog sajta;
- pomoću CDN-a.

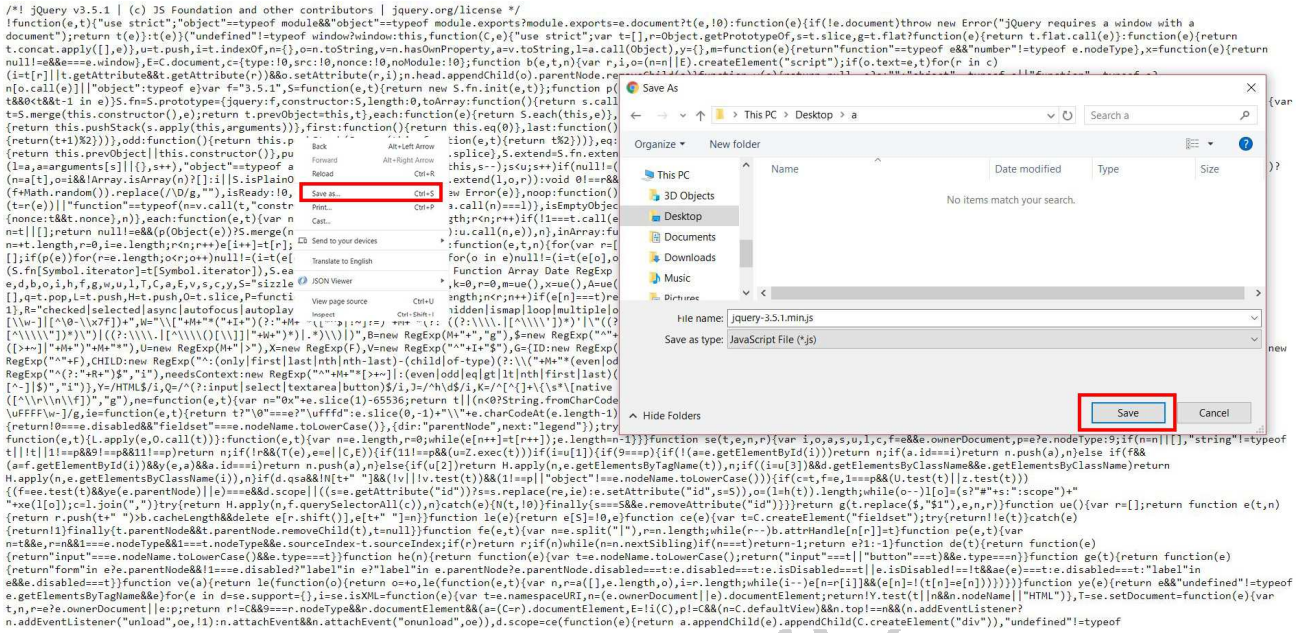
Uvoženje jQuery biblioteke

Prvi način uvoženja jQuery biblioteke u projekat je preuzimanjem fajla sa zvaničnog sajta:

https://jquery.com/download/

Poželjno je uvek preuzeti poslednju verziju (), kako bismo uvek imali uvezenu biblioteku sa poslednjim izmenama. Prvi link na pomenutoj stranici je uvek kompresovana jQuery biblioteka koja je pogodna za korišćenje u ovoj lekciji. Razlika između ove dve datoteke je u tome što je nekompresovana verzija čitljivija, sa puno formatiranog koda i velikim brojem komentara koji mogu pomoći u razumevanju koda, i često se koristi prilikom razvoja aplikacija, dok je prednost kompresovane verzije manja veličina same datoteke, što omogućuje brže učitavanje web sajtova.

Sam fajl se neće automatski preuzeti, već će se prikazati u pregledaču. Potrebno je prikazani kod sačuvati kao jquery.min.js desnim klikom na kod i izborom opcije Save As, a zatim Save. Postupak čuvanja datoteke je prikazan na slici 11.1.



Slika 11.1. Čuvanje jQuery datoteke

Implementacija jQuery biblioteke vrši se pre zatvaranja body taga.

```
<script src=”jquery.min.js”></script>
```

Napomena:

Bitno je voditi računa o putanji navedenoj u src delu, jer od toga zavisi da li će jQuery bibiliotека raditi. U našem primeru se jQuery biblioteka nalazi u istom folderu kao i HTML fajl gde biblioteku uključujemo.

Uvoženje jQuery biblioteke putem CDN-a

Drugi način dodavanja jQuery biblioteke je putem CDN-a. To znači da je potrebno povući biblioteku sa drugog izvora:

https://code.jquery.com/

Na prethodno pomenutom linku se nalazi lista svih jQuery verzija. Predlaže se preuzimanje poslednje verzije, koja se uvek nalazi na vrhu stranice.

Implementacija jQuery biblioteke vrši se pre zatvaranja body taga.

```
<script                src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
```

jQuery sintaksa – selektori

Svako kome je CSS poznat primetiće da je logika jQuery-ja slična logici CSS-a. Prvobitno ćemo se u lekciji baviti selektovanjem elmenata. Baš kao i kod CSS-a, selektovanje se koristi za pronalaženje elemenata na web stranici. Bitni su samo kriterijumi po kojima će se izvšiti selektovanje; neki od najčešće korišćenih su: id, klasa, atribut, tip elementa i dr. Lista svih selektora se možete videti u zvaničnoj dokumentaciji.
Znak **$** se koristi za pristupanje jQuery bibliotici.

Selekovanje elementa prema tipu elementa

Primer 1

```
<!DOCTYPE html>
<html>
<head>
	<title>jQuery - Selektovanje elemenata prema tipu elementa</title>
</head>
<body>
	<p>Text 1</p>
	<p>Text 2</p>
	<p>Text 3</p>

	<script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0=" crossorigin="anonymous"></script>
	<script type="text/javascript">
		$("p").text("New text!");
	</script>
</body>
</html>
```

Kao što je u primeru pokazano, unutar prve zagrade upisujemo atribut koji želimo da selektujemo – ovog puta je to p element. Korišćenjem selektovanja prema tipu elementa, dobićemo sve elemente koji su tog tipa. U našem primeru, izmenićemo tekst unutar svih p elemenata na vrednost *New text!*.

Izmeniti kod u radnom okruženju tako da stranica sadrži 2 prazna naslova veličine h1 i h2 i pomoću jquery-ja se tekst prvog naslova menja u Vaše ime a tekst naslova veličine h2 u prezime.

Selektovanje elemenata na osnovu ID atributa

Najćešće se selektovanje vrši putem ID atributa. Kako bi selektovanje bilo uspešno, bitno je da na strani koju pretražujemo ID bude jedinstven.

Primer 2

Radno okruženje

```
<!DOCTYPE html>
<html>
```

```
<head>
      <title>
         jQuery - Selektovanje elemenata na osnovu ID atributa
      </title>
</head>

<body>
      <div id="text">Text</div>

      <script        src="https://code.jquery.com/jquery-3.5.1.min.js"
      integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
      crossorigin="anonymous"></script>
      <script type="text/javascript">
            $("#text").text("New text!");
      </script>
</body>
</html>
```

U ovom primeru, slično CSS-u, kada selektujemo element putem ID-ja, koristimo znak # i vrednost samog ID atributa. Vrednost unutar p taga koji ima ID atribut text biće promenjena na *New text!*.

Selektovanje po class atributu

Na ovaj način selektuju se sve vrednosti koje imaju class atribut. Ovaj postupak je vrlo sličan selektovanju prema ID atrubutu; razlika je u tome što rezultat može biti više elemenata.

Primer 3

```
<!DOCTYPE html>
<html>
<head>
      <title>jQuery - Selektovanje po class atributu</title>
</head>
<body>
    <p class="Class1">Text 1</p>
    <p class="Class1">Text 2</p>
    <p>Text 3</p>

    <script          src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
    <script type="text/javascript">
      $(".Class1").text("New text!");
    </script>
</body>
</html>
```

U primeru je prikazano selektovanje dva p elementa na osnovu class atrubuta, dok treći p element, koji nema class atribut, nije selektovan i njegova vrednost neće biti promenjena.

Izmenite u radnom okruženju kod tako da se pomoću jquery-ja menja tekst 2 naslova veličine h1 koji imaju vrednost NASLOV na VELIKI NASLOV.

jQuery – upravljanje događajima

Pomoću jQuery biblioteke može se kontrolisati mnogo događaja. Najčešće korišćene metode za upravljanje događajima su prikazane u tabeli:

Upravljanje događajima miša	Upravljanje događajima tastature	Upravljanje događajima nad formom	Upravljanje događajima dokumenta
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Tabela 11.1. Najčešći događaji

Sintaksa za definisanje događaja je slična za sve metode:

```
$(selektor).naziv_metode();
```

Primer 5

```
<!DOCTYPE html>
<html>
<head>
  <title>jQuery - Upravljanje događajima</title>
</head>
<body>
        <p>Text</p>
        <script      src="https://code.jquery.com/jquery-3.5.1.min.js"
      integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
      crossorigin="anonymous"></script>
        <script type="text/javascript">
          $("p").click(function(){
            alert("Click");
          });
        </script>
</body>
</html>
```

Izmeniti kod u radnom okruženju tako da kada se klikne na horizontalnu liniju (hr tag) se u alert prozoru ispisuje današnji datum.

U primeru broj 5 koristimo metodu click, koju vezujemo za p element. Klikom na p element poziva se funkcija koja će unutar prozora za upozorenje (alert) ispisati tekst *Click*.

Primer 6

```
<!DOCTYPE html>
<html>
<head>
  <title>jQuery - Upravljanje događajima</title>
  <style>
    .box {
      background-color: red;
      width: 100px;
      height: 100px;
    }
  </style>
</head>
<body>
 <div class="box"></div>
 <h1>OUT</h1>
 <script             src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
 <script type="text/javascript">
  var div1=$("div").mouseenter(function(){
   $("div").css("background-color","blue");
   $("h1").text("IN");
 })
</script>
</body>
</html>
```

U primeru broj 6 smo nad div elementom definisali događaj mouseenter. Ukoliko korisnik pređe mišem preko spomenutog diva, nad divom će se izvršiti css() metoda koja kao prvi ulazni parametar prima svojstvo, a kao drugi vrednost; u našem slučaju to su svojstvo background-color i vrednost blue. Takođe, unutar funkcije koja će se pozvati prilikom odigravanja događaja menjamo i vrednost h1 elementa u tekst *IN* pomoću text metode.

Izmeniti kod u radnom okruženju tako da kada se miš napusti div odn. Miš se ne nalazi na div-u pozadina diva bude crvena a tekst naslova unutar diva da bude OUT.

Radno okruženje

Često ćemo se naći u situaciji kada je potrebno da određeni deo koda izvršimo tek kada je stranica učitana (kada je DOM učitan). To možemo uraditi pomoću metode ready().

```
$(document).ready(function(){
      //Code...
});
```

Kreiranje novih HTML elemenata

Kako bismo kreirali nove HTML elemente, potrebno je da prođemo kroz dva koraka. Prvi je da kreiramo novi element, dodelimo mu sadržaj i eventualno podesimo atribute, a drugi je da element smestimo na željenu lokaciju u dokumentu. Već smo naučili kako da kreiramo element pomoću HTML-a i DOM-a. Sintaksa je sledeća:

```
var new_p_tag =$(“<p></p>”).text(“New text”);
```

Novokreiranom elementu možemo dodati atribut metodom attr(), koja prima dva ulazna parametra: sam atribut (vrstu atributa) i njegovu vrednost. Elementu new_p_tag dodelićemo atribut id, koji će imati vrednost new_id.

```
var new_p_tag =$(“<p></p>”).text(“New text”);
new_p_tag.attr(“id”,”new_id”);
```

Ovaj atribut još uvek nije vidljiv na stranici, jer još uvek nije postavljen na željeno mesto. Za postavljanje atributa imamo na raspolaganju nekoliko različitih metoda.

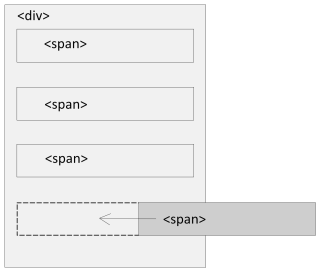
Metode za dodavanje novih elemenata u HTML dokument

Postoje četiri metode za dodavanje novih elemenata unutar HTML stranice:

- append() – dodaje sadržaj na kraj selektovanog elementa;
- prepend() – dodaje sadržaj na početak selektovanog elementa;
- after() – dodaje sadržaj nakon selektovanog elementa;
- before() – dodaje sadržaj pre selektovanog elementa.

Dodavanje sadržaja na kraj selektovanog elementa – append()

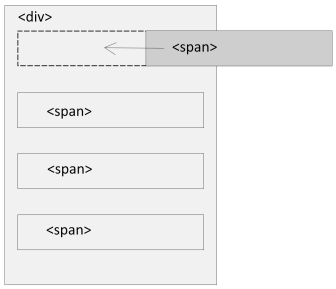
Metoda append() je najčešće korišćena među ovim metodama; ona dodaje element na kraj selektovanog elementa.



Slika 11.2. jQuery append()

Ako kao primer postavimo da želimo da dodamo novi span element u div element koji već ima tri span elementa, novi element će biti postavljen nakon postojeća tri.

Dodavanje sadržaja na početak selektovanog elementa – prepend()

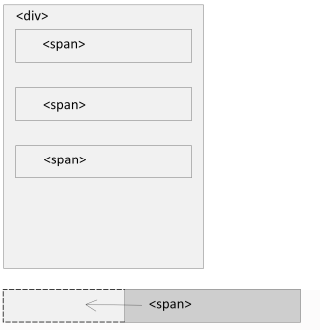


Slika 11.3. jQuery prepend()

Novi element će biti dodat na sam početak, pre postojećih elemenata.

Dodavanje sadržaja nakon selektovanog elementa – after()

Ova metoda ne dodaje novi element selektovanom elementu, već postavlja novi element nakon selektovanog elementa.



Slika 11.4. jQuery after()

Dodavanje sadržaja pre selektovanog elementa – before()

Ova metoda je slična prethodnoj; razlika je u tome što postavlja sadržaj pre selektovanog elementa.

Slika 11.5. jQuery before()

Uklanjanje sadržaja iz selektovanog elementa – empty()

Element div koji smo koristili u prethodnim primerima poseduje tri span elemenata unutar sebe. U slučaju da želimo da mu izmenimo strukturu, potrebno je da prethodno ispraznimo div element, odnosno uklonimo sve njegove elemente (u ovom slučaju – tri span elementa). Iz tog razloga kreirana je posebna empty() metoda, koju možemo pozvati nad elementom.

Primer 7

Radno okruženje

```
<!DOCTYPE html>
<html>
    <head>
        <title>jQuery - method empty()</title>
    </head>
    <body>
         <div>
             <button type="button">Action</button>
             <div id="element">
                 <h1>Header H1</h1>
                 <p>Paragraf 1</p>
                 <p>Paragraf 2</p>
             </div>
         </div>

 <script             src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
        <script type="text/javascript">
            $(document).ready(function(){
            $("button").click(function(){
            $("#element").empty();
            })
         })
        </script>
    </body>
</html>
```

Izmenite u radnom okruženju kod tako da se umesto div-a brišu paragrafi koji se nalaze unutar njega. Po potrebi dodati klasu elementima.

Uklanjanje čitavog elementa – remove()

Ta razliku od metode empty(), koja uklanja samo potomke selektovanog elementa, metoda remove() uklanja i potomke elementa i sam element koji je selektovan.

jQuery – efekti

jQuery biblioteka nam nudi već pripremljene efekte. Jedna od osnovnih metoda za efekte je hide().

Dovoljno je selektovati element i pozvati metodu hide() kako bi efekat bio postignut. Pored metode hide imamo i metodu show(). Lista svih efekata je dostupna u zvaničnoj dokumentaciji.

Primer 8

```
<!DOCTYPE html>
<html>
<head>
  <title>jQuery - show/hide effects</title>
</head>
<body>
  <p>If you click on the "Hide" button, I will disappear.</p>

  <button id="hide">Hide</button>
  <button id="show">Show</button>

  <script            src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>

  <script type="text/javascript">
    $(document).ready(function(){
      $("#hide").click(function(){
        $("p").hide();
      });
      $("#show").click(function(){
        $("p").show();
      });
    });
  </script>
</body>
</html>
```

U primeru je prikazano funkcionisanje metoda hide() i show(). Onog trenutka kada kliknemo na dugme sa ID atributom hide, p element će nestati, a kada kliknemo na dugme sa ID atributom show, p element će se ponovo prikazati.

Izmenite u radnom okruženju kod tako da kada se klikne na dugme Remove se paragram briše sa stranice.

AJAX

Proći ćemo kroz dva primera upotrebe AJAX-a korišćenjem jQuery metoda. Prvi primer šalje zahtev koristeći GET metodu, a drugi koristeći POST metodu.

Implementacije klijenta (JavaScript/jQuery) i servera (Python) prikazaćemo na primeru gde server sadrži rečnik imena i prezimena zvan names_dict, a klijent pokušava ili da dobavi prezime za dato ime (GET) ili da doda nove vrednosti tom rečniku (POST) koristeći upitne stringove.

Pitanje

Koju metodu koristimo kada želimo da uklonimo element?

- empty()
- readyState
- **remove()**
- setRequestHeader()

Objašnjenje:

Tačan odgovor je remove(). Ova metoda, za razliku od metode empty(), koja uklanja samo unutrašnje elemente selektovanog elementa, uklanja kompletan element.

Python server

Za server koristimo već poznati kod iz kursa *Python Net Programming* kursa. Jedina izmena jeste dodavanje linije:

```
self.send_header('Access-Control-Allow-Origin', '*')
```

unutar send_response_to_client() funkcije. Ova linija nam omogućava da server kontaktiramo sa bilo kog klijenta.

Ovu skriptu ćemo sačuvati pod imenom http_local_server.py.

Fajl: http_local_server.py

```
from http.server import HTTPServer, BaseHTTPRequestHandler
from urllib.parse import parse_qs
names_dict = {'john':'smith',
              'david':'jones',
              'michael':'johnson',
              'chris':'lee'}
class RequestHandler(BaseHTTPRequestHandler):
    def do_GET(self):
        self.log_message("Incoming GET request...")
```

```
            try:
                name = parse_qs(self.path[2:])['name'][0]
            except:
                self.send_response_to_client(404,   'Incorrect   parameters
provided')
                self.log_message("Incorrect parameters provided")
                return
            if name in names_dict.keys():
                self.send_response_to_client(200, names_dict[name])
            else:
                self.send_response_to_client(400, 'Name not found')
                self.log_message("Name not found")

        def do_POST(self):
            self.log_message('Incoming POST request...')
            data = parse_qs(self.path[2:])
            try:
                names_dict[data['name'][0]] = data['last_name'][0]
                self.send_response_to_client(200, names_dict)
            except KeyError:
                self.send_response_to_client(404,   'Incorrect   parameters
provided')
                self.log_message("Incorrect parameters provided")

        def send_response_to_client(self, status_code, data):
            # Send OK status
            self.send_response(status_code)
            # Send headers
            self.send_header('Content-type', 'text/plain')
            self.send_header('Access-Control-Allow-Origin', '*')
            self.end_headers()

            # Send the response
            self.wfile.write(str(data).encode())

    server_address = ('127.0.0.1', 8080)
    http_server = HTTPServer(server_address, RequestHandler)
    http_server.serve_forever()
```

Za pokretanje servera potrebno je pokrenuti komandni prozor i pozicionirati se u direktorijum gde se nalazi fajl http_local_server.py. Potrebno je pokrenuti skriptu komandom python http_local_server.py.

GET request

Kroz sledeći primer biće objašnjen GET poziv uz korišćenje JQuery metode get(). Metoda ima dva parametra: jedan je string koji je ujedno i željeni URL ka serveru, a drugi je funkcija koja se izvršava odmah pri dospeću odgovora od servera. Funkcija prima prvi parametar, koji je odgovor sa servera.

Napomena:

Kako bismo kroz sam URL prosledili ime iz input polja, potrebno je da nad selektovanim poljem $(“#name”) pokupimo unetu vrednost metodom val().

Nakon preuzimanja parametra, ispisivanje se vrši preko metode text(), koja ispisuje u p element vrednost koju smo zadali, a to je u ovom slučaju odgovor sa servera.

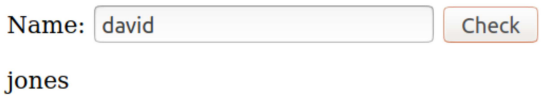
Primer 8

```
<!DOCTYPE html>
<html>
<head>
<title>jQuery - AJAX GET method</title>
</head>
<body>
  <div>
    <label>Name:</label>
    <input id="name">
    <button onclick="checkForName()">Check</button>
    <br>
    <p id="response"></p>
  </div>


  <script            src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>

  <script>
   function checkForName() {
    var name = $('#name').val();
    $.get("http://127.0.0.1:8080/?name=" + name, function (data) {
      $("#response").text(data);
    }).fail(function(err){
      $("#response").text(err.responseText);
    });
  }
</script>
</body>
</html>
```

Iz primera možemo videti da, nakon što je server vratio korisniku odgovor koji se nalazi unutar promenljive data, isti odgovor ispisujemo unutar p elementa koji ima vrednost ID atributa response. Naš Python server je podešen tako da, ukoliko ime ne postoji unutar rečnika, vrati grešku, odnosno status 400. Zbog toga je neophodno da tu grešku uhvatimo lančanom funkcijom fail(). Funkcija prima prvi parametar, objekat, koji sadrži informacije o poslatom zahtevu i primljenom odgovoru sa servera, gde možemo videti informacije poput tekstualnog odgovora, statusa koji je server vratio i slično.

Na sledećim slikama je prikazano kako izgleda stranica pre poziva na server i posle poziva.

Slika 11.6. Prikaz stranice pre slanja GET zahteva



Slika 11.7. Prikaz stranice nakon slanja GET zahteva i ispis rezultata

POST request

POST se veoma malo razlikuje od GET zahteva. Jedina razlika je u tome što se koristi post() metoda.

Dodat je još jedan input element kako bi bio ispunjen zahtev url-a, a to je da se pošalje ime i prezime.

Kada se izvršio uspešan zahtev, dobili smo odgovor koji smo prikazali u konzoli pregledača.

Primer 9

```
<!DOCTYPE html>
<html>
<head>
  <title>jQuery - AJAX POST method</title>
</head>
<body>
  <div>
    <label>Name:</label>
    <input id="name">
    <br>
    <label>Surname:</label>
    <input id="surname">
    <br>
    <button onclick="addNew()">Get</button>

  </div>


  <script            src="https://code.jquery.com/jquery-3.5.1.min.js"
integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>

  <script>
   function addNew() {
    var name = $('#name').val();
```

```
        var surname = $('#surname').val();
        $.post("http://127.0.0.1:8080/?name="  +  name  +  ";last_name="  +
    surname, function (data) {
          console.log(data);
        });
      }

    </script>
    </body>
    </html>
```

Slika 11.8. Prikaz stranice pre slanja POST zahteva

Kao odgovor sa servera dobijamo rezultat u sledećem formatu:

```
{'john': 'smith', 'david': 'jones', 'michael': 'johnson', 'chris': 'lee',
'sonja': 'elton'}
```

Obeležen kod, odnosno ključ '**sonja**' i vrednost '**elton**', jesu vrednosti koje je korisnik uneo putem input polja.

Rezime

- Implementacija jQuery biblioteke vrši se na dva načina: lokalno i korišćenjem CDN-a.
- Za odabir elemenata se može koristiti veliki broj selektora.
- Za dodavanje elemenata najčešće se koriste metode append(), prepend(), after() i before().
- Za uklanjanje elemenata koristimo metode empty() i remove().
- Za slanje asinhronih zahteva ka serveru možemo koristi jQuery get() i post() metode.